

# GENERIC CASE COMPLETENESS

ALEXEI MIASNIKOV AND ALEXANDER USHAKOV

**ABSTRACT.** In this note we introduce a notion of a generically (strongly generically) **NP**-complete problem and show that the randomized bounded version of the halting problem is strongly generically **NP**-complete.

**Keywords.** Generic-case complexity, completeness, randomized problems, bounded halting problem.

**2010 Mathematics Subject Classification.** 68Q17.

## CONTENTS

1. Introduction	1
2. Preliminaries	3
2.1. Decision problems	3
2.2. Deterministic and nondeterministic Turing machines	4
2.3. Polynomial time reductions	5
3. Distributional problems and generic case complexity	6
3.1. Distributional decision problems	6
3.2. Generic complexity	7
3.3. Distributional <b>NP</b> -problems	8
4. Generic Ptime reductions	8
4.1. Change of size	9
4.2. Change of measure	10
4.3. Reduction to a problem with the binary alphabet	11
4.4. On restrictions of problems	12
5. Distributional bounded halting problem	13
6. Open problems	18
References	18

## 1. INTRODUCTION

We introduce and study problems that are generically in **NP**, i.e., decision problems that have partial errorless nondeterministic decision algorithms that solve the problem in polynomial time on “most” inputs. We define appropriate reductions in this class and show that there are some complete problems there, called *strongly generically NP-complete* problems. In particular, the randomized bounded version of the halting problem is one of them.

Rigorous formulation of notions of generic algorithms and generic complexity appeared first in group theory [17, 18] as a response to several challenges that

---

The work was partially supported by NSF grant DMS-1318716.

algorithmic algebra faced at that time. First, it was well understood that many hard, even undecidable, algorithmic problems in groups can be easily solved on most instances (see [17, 18, 8, 21] for a thorough discussion). Second, the study of random objects and generic properties of objects has become the mainstream of geometric group theory, following the lead of graph and number theory (see [9, 10, 11, 23, 1, 4, 3]). It turned out that “random”, “typical” objects have many nice properties that lead to simple and efficient algorithms. However a rigorous formalization of this approach was lagging behind. Algorithmic algebra was still focusing mostly on the worst-case complexity with minor inroads into average case complexity. Third, with the rapid development of algebraic cryptography the quest for natural algorithmic problems, which are hard on most inputs, became one of the main subjects in complexity theory (see discussion in [21]). It was realized that the average case complexity does not fit well here. Indeed, by definition, one cannot consider average case complexity of undecidable problems, which are in the majority in group theory; the proofs of average case results are usually difficult and technical [12, 25], and, most importantly, there are problems that are provably hard on average but easy on most inputs (see [8, 21] for details). In fact, Gurevich showed in [12] that the average case complexity is not about “most” or “typical” instances, but that it grasps the notion of “trade-off” between the time of computation on hard inputs and how many of such hard instances are there. Nowadays, generic algorithms form an organic part of computational algebra and play an essential role in practical computations.

In a surprising twist generic algorithms and ideas of generic complexity were recently adopted in abstract computability (recursion theory). There is interesting and active research there concerning absolutely undecidable problems, generic Turing degrees, coarse computability, etc., relating generic computation with deep structural properties of Turing degrees [20, 16, 2, 14, 6, 5].

We decided to relativize these ideas to lower complexity classes. Here we consider the class **NP**. Motivation to study generically hardest problems in the class **NP** comes from several areas of mathematics and computer science. First, as we have mentioned above, average case complexity, even when it is high, does not give information on the hardness of the problem at hand on the typical or generic inputs. Therefore, to study hardness of the problem on most inputs one needs to develop a theory of generically complete problems in the class **NP**. This is interesting in its own right, especially when much of activity in modern mathematics focuses on generic properties of mathematical objects and how to deal with them. On the other hand, in modern cryptography, there is a quest for cryptoprimitives which are computationally hard to break on most inputs. It would be interesting to analyze which **NP**-problems are hard on most inputs, i.e., which of them are generically **NP**-complete. Note, there are **NP**-complete problems that are generically polynomial [21]. All this requires a robust theory of generic **NP**-completeness. As the first attempt to develop such a theory we study here the class of all generically **NP**-problems, their reductions, and the complete problems in the class. Most of the time, our exposition follows the seminal Gurevich’s paper [12] on average complexity. We conclude with several open problems that seem to be important for the theory.

Here we briefly describe the structure of the paper and mention the main results. In Section 2, we recall some notions and introduce notation from the classical decision problems. In Section 3, we discuss distributional decision problems (when the set of instances of the problem comes equipped with some measure), then define the generic complexity and problems decidable generically (strongly generically) in polynomial time. In Section 4, we define generic polynomial time reductions. In Section 5, we show that the distributional bounded halting problem for Turing machines is strongly generically **NP**-complete. Notice that though generic Ptime randomized algorithms are usually much easier to come up with (than say Ptime on average algorithms), the reductions in the class of generic **NP**-problems are still as technical as reductions in the class of **NP**-problems on average. In fact, the reductions in both classes are similar. Essentially, these are reductions among general randomized problems and the main technical, as well as theoretical, difficulty concerns the transfer of the measure when reducing one randomized problem to another one. It seems this difficulty is intrinsic to reductions in randomized computations and does not depend on whether we consider generic or average complexity. In Section 6 we discuss some open problems that seem to be important for the development of the theory of generic **NP**-completeness.

## 2. PRELIMINARIES

In this section we introduce notation to follow throughout the paper.

**2.1. Decision problems.** Informally, a *decision problem* is an arbitrary yes-or-no question for an (infinite) set of *inputs* (or *instances*)  $I$ , i.e., an unary predicate  $P$  on  $I$ . The problem is termed *decidable* if  $P$  is computable, and the main classical question is whether a given problem is decidable or not. In complexity theory the predicate  $P$  usually is given by its true set  $L = \{x \in I \mid P(x) = 1\}$ , so the decision problem appears as a pair  $(I, L)$ . Furthermore, it is assumed usually that every input  $x \in I$  admits a finite description in some finite alphabet  $\Sigma$  in such a way that given a word  $w \in \Sigma^*$  one can effectively determine if  $w \in I$  or not. This allows one, without loss of generality, to assume simply that  $I = \Sigma^*$ . Some care is required when dealing with distributional problems and we discuss this issue in due course. From now on, unless said otherwise, we assume that decision problems are pairs  $D = (\Sigma^*, L)$ , where  $L \subseteq \Sigma^*$ . In this case  $\Sigma$  is the *alphabet* of the problem  $D$  and we denote it sometimes by  $\Sigma_D$ ;  $\Sigma_D^*$  is the set of *inputs* or the *domain* of  $D$ ; the set  $L$  is the *yes* or *positive* part of  $D$ , denoted sometimes by  $D^{yes}$  or  $D^+$ . In Section 4.4 we briefly consider problems of the type  $(I, L)$ , where  $L \subseteq I \subseteq \Sigma^*$ , not assuming that  $I$  is a decidable subset of  $\Sigma^*$ .

It is natural now to define the *size* of  $x \in \Sigma^*$  to be its word length  $|x|$ . As usual, we define the sphere  $\Sigma^n$  of radius  $n \in \mathbb{N}$  as the set of all strings (words) in  $\Sigma^*$  of size  $n$ , and  $D_n = D \cap \Sigma^n$ . For a symbol  $a \in \Sigma$  and  $n \in \mathbb{N}$  put  $a^n$  to be the string of  $n$  symbols  $a$ .

We assume that alphabet  $\Sigma$  comes equipped with a fixed linear ordering. This allows one to introduce a *shortlex* ordering  $<_{slex}$  on the set  $\Sigma^*$  as follows. We order, first, the words in  $\Sigma^*$  with respect to their length (size), and if two words have the same length then we compare them in the (left) lexicographical ordering. The successor of a word  $x \in \Sigma^*$  is denoted by  $x^+$ .

**2.2. Deterministic and nondeterministic Turing machines.** In this section we recall the definition of a Turing machine in order to establish terminology.

**Definition 2.1.** A one-tape *Turing machine* (TM)  $M$  is a 5-tuple  $\langle Q, \Sigma, q_0, q_1, \delta \rangle$  where:

- $Q = \{q_0, q_1, \dots, q_m\}$  is a finite set of *states*;
- $\Sigma = \{a_1, \dots, a_n\}$  is a finite set called the *tape alphabet* which contains at least 2 symbols;
- $q_0 \in Q$  is the *initial state*;
- $q_1 \in Q$  is the *final state*;
- $\delta \subset Q \times (\Sigma \cup \{\sqcup\}) \times Q \times \Sigma \times \{L, R\}$  is the *transition relation*.

Additionally,  $M$  uses a *blank symbol*  $\sqcup$  different from the symbols  $\Sigma$  to mark the parts of the infinite tape not in use. This is the only symbol allowed to occur on the tape infinitely often at any step during the computation.

We say that a transition relation  $\delta$  in the definition of a TM is *deterministic* if for every pair  $(q, a) \in Q \times (\Sigma \cup \{\sqcup\})$  there is a unique five-tuple  $(q, a, q', \gamma', d)$  in  $\delta$ , i.e.,  $\delta$  defines a function  $\delta^* : Q \times (\Sigma \cup \{\sqcup\}) \rightarrow Q \times \Sigma \times \{L, R\}$ . We say that a TM  $M$  is *deterministic* if its transition relation is. Otherwise we say that  $M$  is a *nondeterministic machine* (NTM).

Each Turing machine has a *tape* with  $(\Sigma \cup \{\sqcup\})$ -symbols written on it, a *head* specifying a position on the tape, and a *state register* containing an element  $q \in Q$ . We say that the head observes a symbol  $a \in \Sigma \cup \{\sqcup\}$ , if  $a$  is written on the tape at the position specified by the head. If a TM  $M$  is in the state  $q$  and observes a symbol  $a \in \Sigma$ , then to perform a step of computations:

- $M$  chooses any element  $(q, a, q', a', d) \in \delta$ ;
- puts  $q'$  into the state register;
- writes  $a'$  on the tape to the head position;
- moves the head to left or to the right depending on  $d$ .

If  $\delta$  contains no tuple  $(q, a, q', a', d)$ , then we say that  $M$  *breaks*.

We can define the operation of a TM formally using the notion of a configuration that contains a complete description of the current state of computation. A *configuration* of  $M$  is a triple  $(q, w, u)$  where  $w, u$  are  $\Sigma$ -strings and  $q \in Q$ .

- $w$  is a string to the left of the head;
- $u$  is the string to the right of the head, including the symbol scanned by the head;
- $q$  is the current state.

We say that a configuration  $(q, w, u)$  *yields* a configuration  $(q', w', u')$  in one step, denoted by

$$(q, w, u) \xrightarrow{M} (q', w', u'),$$

if a step of a machine from configuration  $(q, w, u)$  results in configuration  $(q', w', u')$ . Note that if the machine is nondeterministic, then a configuration can yield more than one configuration. Using the relation “yields in one step” one can define relations “yields in  $k$  steps”, denoted by

$$(q, w, u) \xrightarrow{M^k} (q', w', u'),$$

and “yields”, denoted by

$$(q, w, u) \xrightarrow{M^*} (q', w', u').$$

We say that  $M$  *halts* on  $x \in \Sigma^*$  if the configuration  $(q_0, \varepsilon, x)$  yields a configuration  $(q_1, w, u)$  for some  $\Sigma$ -strings  $w$  and  $u$ . The number of steps  $M$  takes to stop on a  $\Sigma$ -string  $x$  is denoted by  $T_M(x)$ . If  $M$  does not halt on  $x \in \Sigma^*$  then we put  $T_M(x) = \infty$ .

The *halting problem* for  $M$  is an algorithmic question to determine whether  $M$  halts or not on an input  $x \in \Sigma^*$ , i.e., whether  $T_M(x) = \infty$  or not.

We say that a TM  $M$  *solves* or *decides* a decision problem  $D$  over an alphabet  $\Sigma$  if  $M$  stops on every input  $x \in \Sigma^*$  with an answer:

- *Yes* (i.e., at a configuration  $(f, \varepsilon, w)$ , where  $w$  starts with  $a_1 a_1$ ) if  $x \in L(D)$ ;
- *No* (i.e., at configuration  $(f, \varepsilon, w)$ , where  $w$  starts with  $a_1 a_0$ ) otherwise.

We say that  $M$  *partially decides*  $D$  if it decides  $D$  correctly on a subset  $D'$  of  $D$  and on  $D - D'$  it either does not stop or stops with an answer *DontKnow* (i.e., stops at configuration  $(f, \varepsilon, w)$ , where  $w$  starts with  $a_0$ ). In the event when  $M$  breaks or outputs *DontKnow* the value of  $T_M(x)$  is  $\infty$ .

**2.3. Polynomial time reductions.** For a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  define **TIME**( $f$ ) [**NTIME**( $f$ ) resp.] to be the class of all decision problems decidable by some deterministic [nondeterministic resp.] Turing machine within time  $f(n)$ . Two of the most used classes of decision problems **P** and **NP** are defined as follows:

$$\mathbf{P} = \bigcup_{k=1}^{\infty} \mathbf{TIME}(n^k) \quad \text{and} \quad \mathbf{NP} = \bigcup_{k=1}^{\infty} \mathbf{NTIME}(n^k).$$

Clearly  $\mathbf{P} \subseteq \mathbf{NP}$ . It is an old, open problem whether  $\mathbf{NP} = \mathbf{P}$  or not.

The classical *polynomial time many-to-one* or *Karp reductions* provide a crucial tool to deal with problems in **NP**. We recall it in the following definition and refer to them simply as to Ptime reductions.

**Definition 2.2.** Let  $D_1$  and  $D_2$  be decision problems. We say that a function  $f : \Sigma_{D_1}^* \rightarrow \Sigma_{D_2}^*$  is a *Ptime reduction*, or  $f$  *Ptime reduces*  $D_1$  to  $D_2$ , and write  $D_1 \xrightarrow{f}_P D_2$ , if

- $f$  is polynomial time computable;
- $x \in D_1$  if and only if  $f(x) \in D_2$ .

We say that a Ptime reduction  $f$  is *size-invariant* if

$$|x_1| < |x_2| \iff |f(x_1)| < |f(x_2)|.$$

Notice, that many classical Ptime reductions are size-invariant (see [24]).

Now, for a size-invariant reduction  $f$  the function

$$\mathcal{S}_f(n) := |f(x)|, \text{ where } |x| = n,$$

is well defined and strictly increasing. We refer to  $\mathcal{S}_f$  as the *size growth* of  $f$ .

A problem  $D \in \mathbf{NP}$  is called **NP-complete** if every problem  $D' \in \mathbf{NP}$  is Ptime reducible to  $D$ . The following is a classic result in complexity theory (see [24]).

**Theorem 2.3.** *The following holds.*

- (a) *If  $f$  is a Ptime reduction from  $D_1$  to  $D_2$  and  $M$  is an Turing machine solving  $D_2$  in polynomial time then  $M \circ f$  solves  $D_1$  in polynomial time.*
- (b) *3SAT is NP-complete.*

Here, and below, by  $M \circ f$  we denote the algorithm that is a composition of the TM  $M$  and a TM that computes  $f$ .

### 3. DISTRIBUTIONAL PROBLEMS AND GENERIC CASE COMPLEXITY

Let us first recall some basic definitions of probability theory that will be used in this section. A *probability measure* on  $\Sigma^*$  is a function  $\mu : \Sigma^* \rightarrow [0, 1]$  satisfying  $\sum_{x \in \Sigma^*} \mu(x) = 1$ . An *ensemble of probability measures* on  $\Sigma^*$  is a collection of sets  $\{S_n\}_{n=1}^\infty$  of  $\Sigma^*$  (not necessarily disjoint) and a collection of probability measures  $\mu = \{\mu_n\}_{n=1}^\infty$  satisfying  $\text{supp}(\mu_n) \subseteq S_n$  and  $S = \bigcup S_n$ . A *spherical ensemble of probability measures* on  $\Sigma^*$  is an ensemble with  $S_n = \Sigma^n$ . In particular, a spherical ensemble of probability measures on  $\Sigma^*$  is uniquely defined by a collection of measures  $\{\mu_n\}_{n=1}^\infty$  satisfying  $\text{supp}(\mu_n) \subseteq \Sigma^n$ .

**3.1. Distributional decision problems.** The average case complexity deals with “expected” running time of algorithms, while the generic case complexity deals with “most typical” or *generic* inputs of a given problem  $D = (\Sigma^*, D^+)$ . These require to measure or compare various subsets of inputs from  $\Sigma^*$ . There are several standard ways to do so, for example, by introducing either a probability measure  $\mu$  on  $\Sigma^*$  (as was done in [19, 12]), or an *ensemble* of probability measures defined on spheres or balls of  $\Sigma^*$  (see [15]). In many cases, all three approaches are equivalent and lead to similar results. Following the current tradition in computer science, we elect here to work with a *spherical ensemble*  $\mu = \{\mu_n\}_{n=1}^\infty$  of probability measures  $\mu_n$  defined on the spheres  $\Sigma^n$ . In what follows, we always assume that  $\Sigma$  is a finite alphabet and every measure  $\mu_n$  from the ensemble  $\mu$  is atomic, i.e., it is given by a probability function (which we denote again by  $\mu_n$ )  $\mu_n : \Sigma^n \rightarrow \mathbb{R}$  so that  $\mu_n(S) = \sum_{x \in S} \mu_n(x)$  for every subset  $S \subseteq \Sigma^n$ . The pair  $(\Sigma^*, \mu)$  is termed a *distributional space*. Whether  $\mu_n$  is a probability measure or the corresponding probability function will be always clear from the context, so no confusion should arise.

We want to stress here that generic properties of a given decision problem depend on the chosen ensemble  $\mu$  and  $\mu$  is an essential part of the problem (see [12] for details).

**Definition 3.1.** A *distributional decision problem* is a triple  $(\Sigma^*, D^+, \mu)$ , where  $D = (\Sigma^*, D^+)$  is a decision problem and  $(\Sigma^*, \mu)$  a distributional space.

Usually we refer to a distributional problem  $(\Sigma^*, D^+, \mu)$  as a pair  $(D, \mu)$ , where  $D = (\Sigma^*, D^+)$ .

There are two important constructions on distributional spaces, introduced in [12]. Since we use here ensembles of distributions, unlike [12], where single measures were used, we give below precise definitions. Notice, that we always assume that  $\sum_{i \in J} a_i = 0$  if  $J = \emptyset$ .

**Definition 3.2** (Transfers of ensembles). Let  $\Sigma$  and  $\Pi$  be finite alphabets,  $(\Sigma^*, \mu)$  and  $(\Pi^*, \nu)$  distributional spaces, and  $f : \Sigma^* \rightarrow \Pi^*$  a size-invariant function. Then  $\nu$  is the *f-transfer* of  $\mu$  (or  $f$  transfers  $\mu$  to  $\nu$ ) if for any  $y \in \Pi^*$  the following equality holds

$$(1) \quad \nu_{|y|}(y) = \begin{cases} \sum_{x \in f^{-1}(y)} \mu_{|x|}(x), & \text{if } |y| = |f(z)| \text{ for some } z; \\ |\Pi|^{-|y|}, & \text{otherwise.} \end{cases}$$

**Definition 3.3** (Induced ensembles). Let  $(\Sigma^*, \mu)$  be a distributional space and  $S \subseteq \Sigma^*$ . Then an ensemble  $\mu^S = \{\mu_n^S\}_{n=1}^\infty$  on  $\Sigma^*$  is called *S-induced* by  $\mu$  if for any

$x \in \Sigma^*$  the following equality holds

$$(2) \quad \mu_{|x|}^S(x) = \begin{cases} \frac{\mu_{|x|}(\{x\} \cap S)}{\mu_{|x|}(S \cap \Sigma^{|x|})}, & \text{if } \mu_{|x|}(S \cap \Sigma^{|x|}) \neq 0; \\ \mu_{|x|}(x), & \text{otherwise.} \end{cases}$$

**3.2. Generic complexity.** Let  $(\Sigma^*, \mu)$  be a distributional space and  $S \subset \Sigma^*$ . The function

$$n \mapsto \mu_n(S \cap \Sigma^n)$$

is called the *density function* of  $S$  and its limit (if exists)

$$\rho(S) = \lim_{n \rightarrow \infty} \mu_n(S \cap \Sigma^n)$$

is called the *asymptotic density* of  $S$  in  $(\Sigma^*, \mu)$ .

**Definition 3.4.** A subset  $S \subseteq \Sigma^*$  is called

- *generic* in  $\Sigma^*$  if  $\rho(S) = 1$ ;
- *strongly generic* in  $\Sigma^*$  if  $\rho(S) = 1$  and  $\mu_n(S \cap \Sigma^n)$  converges to 1 super polynomially fast, i.e.,  $|1 - \mu_n(S \cap \Sigma^n)| = O(n^{-k})$  for any  $k \in \mathbb{N}$ ;
- *negligible* in  $\Sigma^*$  if  $\rho(S) = 0$ ;
- *strongly negligible* in  $\Sigma^*$  if  $\rho(S) = 0$  and  $\mu_n(S \cap \Sigma^n)$  converges to 0 super polynomially fast, i.e.,  $|\mu_n(S \cap \Sigma^n)| = O(n^{-k})$  for any  $k \in \mathbb{N}$ .

Notice that we use the term “generic” in the sense of “typical”. The same term has also been used in complexity and set theory to refer to sets that are far from typical, that are constructed through Cohen forcing.

**Definition 3.5.** Let  $(D, \mu)$  be a distributional decision problem.

- We say that  $(D, \mu)$  is *decidable generically in polynomial time* (or *GPtime decidable*) if there exists a Turing machine  $M$  that partially decides  $D$  within time  $T_M(x)$  and a polynomial  $p(x)$  such that

$$\mu_n\{x \in \Sigma^n \mid T_M(x) > p(n)\} = o(1).$$

In this case we say that  $M$  is a generic polynomial time decision algorithm for  $D$  and  $D$  has *generic time complexity* at most  $p(n)$ .

- We say that  $(D, \mu)$  is *decidable strongly generically in polynomial time* (or *SGPtime decidable*) if there exists a Turing machine  $M$  that partially decides  $D$  within time  $T_M(x)$  and a polynomial  $p(x)$  such that for any polynomial  $q(n)$

$$\mu_n\{x \in \Sigma^n \mid T_M(x) > p(n)\} = o(1/q(n)).$$

In this case, we say that  $M$  is a strongly generic polynomial time decision algorithm for  $D$  and  $D$  has *strong generic time complexity* at most  $p(n)$ .

We refer to the sequence  $\mu_n\{x \in \Sigma^n \mid T_M(x) > p(n)\}$  as a *control sequence* of the algorithm  $M$  relative to the complexity bound  $p$  and denote it by  $\mathcal{C}_{M,p}$ .

In other words, a problem  $(D, \mu)$  is GPtime (SGPtime) decidable if there exists a polynomial time TM that partially decides  $D$  and its halting set is generic (strongly generic) in  $(\Sigma^*, \mu)$ .



**3.3. Distributional NP-problems.** In this section we recall the notion of a distributional **NP**-problem, which is a distributional analog of the classical **NP**-problems.

**Definition 3.6** (Ptime computable real-valued function). A function  $f : \Sigma^* \rightarrow [0, 1]$  is *computable in polynomial time* if there exists a polynomial time algorithm that for every  $x \in \Sigma^*$  and  $k \in \mathbb{N}$  computes a binary fraction  $f_{x,k}$  satisfying

$$|f(x) - f_{x,k}| < 2^{-k}.$$

**Definition 3.7** (Ptime computable ensembles of probability measures). We say that a spherical ensemble of measures  $\mu = \{\mu_n\}_{n=1}^\infty$  on  $\Sigma^*$  is Ptime computable if the function  $\Sigma^* \rightarrow [0, 1]$  defined by  $x \rightarrow \mu_{|x|}(x)$  is Ptime computable.

Denote by  $\mu^* = \{\mu_n^*\}_{n=1}^\infty$  the ensemble of probability *distributions* defined by

$$\mu_{|x|}^*(x) = \mu_{|x|}\left(\{y \in \Sigma^{|x|} \mid y <_{lex} x\}\right).$$

As above, the ensemble  $\mu^*$  is called Ptime computable if the function  $x \rightarrow \mu_{|x|}^*(x)$  is Ptime computable.

**Lemma 3.8.** *Let  $(\Sigma^*, \mu)$  be a distributional space. Then the following hold:*

- (a) *If  $\mu^*$  is Ptime computable then  $\mu$  is Ptime computable.*
- (b) *If  $S$  is a subset of  $\Sigma^*$  such that the function  $n \rightarrow \mu_n(S \cap \Sigma^n)$  is Ptime computable then the  $S$ -induced on  $\Sigma^*$  ensemble of measures  $\mu^S$  is Ptime computable.*

*Proof.* Follows directly from definitions. □

**Definition 3.9.** **DistNP** is a class of distributional decision problems  $(D, \mu)$  such that

- $D \in \mathbf{NP}$ ;
- $\mu^*$  is a Ptime computable ensemble of probability distributions on  $\Sigma_D^*$ .

**Definition 3.10.** **GP** is the class of GPtime decidable distributional decision problems (not necessarily from **DistNP**). **SGP** is the class of SGPtime decidable distributional decision problems.

We want to point out that classes **GP** and **SGP** contain some exotic problems, e.g., some undecidable problems. For more information see [13, 8, 20].

#### 4. GENERIC PTIME REDUCTIONS

In this section we introduce the notion of a *generic polynomial reduction* and describe two particular types of reductions, called size and measure reductions.

Observe first that the classical Karp reductions do not work for generic complexity. Indeed, the following example shows that a Ptime reduction  $D \xrightarrow{f} E$  and a generic polynomial time decision algorithm for  $E$  do not immediately provide a generic polynomial time decision algorithm for  $D$ .

**Example 4.1.** Let  $\Sigma = \{0, 1\}$  be a binary alphabet and  $\mu$  the spherical ensemble of uniform measures  $\mu_n$  on  $\Sigma^n$ . Let  $f : \Sigma^* \rightarrow \Sigma^*$  be a monoid homomorphism defined by

$$0 \xrightarrow{f} 00 \text{ and } 1 \xrightarrow{f} 1.$$



Now, for a decision problem  $D = (\Sigma^*, D^+)$  consider a decision problem  $f(D) = (\Sigma^*, f(D^+))$ . It follows from the construction that  $D \xrightarrow{f} f(D)$  is a Ptime reduction and  $f(D) \in \mathbf{NP}$ , provided  $D \in \mathbf{NP}$ . Furthermore, it is easy to check that the set  $f(\Sigma^*)$ , as well as  $f(D^+)$ , is strongly negligible in  $(\Sigma^*, \mu)$ . This implies that a partial algorithm  $A$  that on an each input from  $\Sigma^* \setminus f(\Sigma^*)$  says “No” and does halt on  $f(\Sigma^*)$ , is a strongly generic polynomial time decision algorithm for  $(f(D), \mu)$ . Nevertheless,  $A$  does not reveal any useful information on  $D$ .

**Definition 4.2.** Let  $(\Sigma^*, D, \mu), (\Delta^*, E, \nu) \in \mathbf{DistNP}$  and  $D \xrightarrow{f} E$  a Ptime size-invariant reduction.

- (R0) We say that  $f$  is a *weak Gptime reduction* if there exists a TM  $M$ , which Gptime decides  $(E, \nu)$  and  $M \circ f$  Gptime decides  $(D, \mu)$ .
- (R1) We say that  $f$  is an *Gptime reduction* if for every TM  $M$ , which Gptime decides  $(E, \nu)$  the composition  $M \circ f$  Gptime decides  $(D, \mu)$ .
- (R2) We say that  $f$  is an *SGptime reduction* if for every TM  $M$ , which SGptime decides  $(E, \nu)$  the composition  $M \circ f$  SGptime decides  $(D, \mu)$ .

We give examples of SGptime reductions in the next two sections.

**Remark 4.3.** One can introduce reductions  $D \xrightarrow{f} E$  of a more general type by allowing the function  $f$  to be defined only on a generic (strongly generic) subset  $Y$  of  $\Sigma_D^*$  with the polynomial time computable characteristic function  $\chi_Y$ .

**Proposition 4.4** (Transitivity of Gptime and SGptime reductions). The classes of all Gptime and SGptime reductions are closed under composition.

*Proof.* Follows directly from the definitions.  $\square$

It is not known if the class of weak Gptime reductions is transitive.

**Definition 4.5.** Let  $(D, \mu)$  be a distributional decision problem. We say that

- $(D, \mu)$  is *SGptime hard* for  $\mathbf{DistNP}$  if every  $\mathbf{DistNP}$  problem SGptime reduces to  $(D, \mu)$ .
- $(D, \mu)$  is *SGptime complete* for  $\mathbf{DistNP}$  if  $(D, \mu) \in \mathbf{DistNP}$  and  $(D, \mu)$  is *SGptime hard* for  $\mathbf{DistNP}$ .

4.1. **Change of size.** In this section, we introduce *change of size* (CS) reductions.

**Definition 4.6.** Let  $(D, \mu), (E, \nu) \in \mathbf{DistNP}$  and  $D \xrightarrow{f} E$  a Ptime size-invariant reduction of  $D$  to  $E$ . If  $\nu$  is the  $f$ -transfer by  $\mu$  (see Section 3.1) then  $f$  is called a *CS-reduction*.

**Theorem 4.7** (CS-reductions are Gptime and SGptime reductions). *Let  $(D, \mu), (E, \nu) \in \mathbf{DistNP}$  and  $(D, \mu) \xrightarrow{f} (E, \nu)$  a CS-reduction. If  $\mathcal{S}_f$  is bounded by a polynomial, then  $f$  is a Gptime and SGptime reduction.*

*Proof.* Let  $A$  be an algorithm that generically decides  $(E, \nu)$  within a polynomial time upper bound  $p(m)$ . Then  $A \circ f$  is a partial decision algorithm for  $(D, \mu)$ . Since  $\nu$  is induced by  $\mu$ , one has:

$$\begin{aligned} o(1) &= \nu_{\mathcal{S}_f(k)} \{f(x) \mid T_A(f(x)) > p(\mathcal{S}_f(k)), |f(x)| = \mathcal{S}_f(k)\} \\ &= \mu_k \{x \mid T_A(f(x)) > p(\mathcal{S}_f(k)), |x| = k\} \\ &\leq \mu_k \{x \mid T_{A \circ f}(x) > p(\mathcal{S}_f(k)) + T_f(k), |x| = k\}. \end{aligned}$$

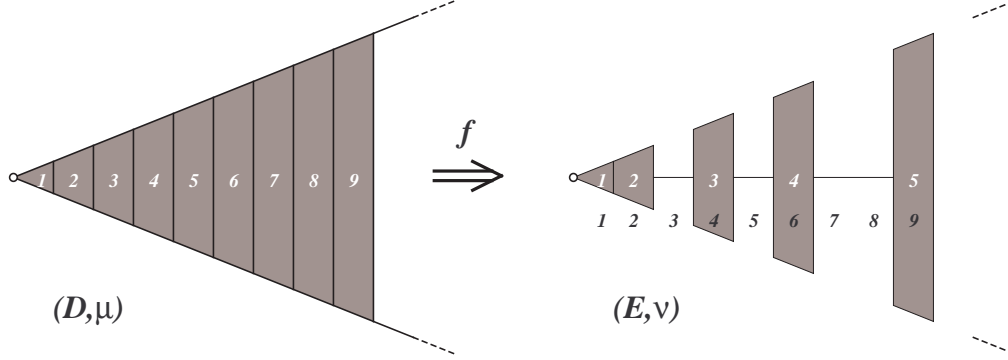


FIGURE 1. In this example  $(E, \nu)$  is obtained from  $(D, \mu)$  by increasing the sizes of elements.

Observe, that  $p \circ \mathcal{S}_f + T_f$  is polynomially bounded, since  $\mathcal{S}_f$  and  $T_f$  are polynomially bounded. Clearly, the control sequence  $\mathcal{C}_{A \circ f, p \circ \mathcal{S}_f + T_f}$  is at most  $\mathcal{C}_{A, p} \circ \mathcal{S}_f$ . Notice, that  $\mathcal{C}_{A, p} \circ \mathcal{S}_f$  is an infinite subsequence of  $\mathcal{C}_{A, p}$  (because  $\mathcal{S}_f$  is strictly increasing), hence it converges to 0, so  $p \circ \mathcal{S}_f + T_f$  is a generic upper bound for  $A \circ f$ . This proves the first statement of the theorem.

To prove the second statement, assume that  $(E, \nu)$  is SGPTIME decidable by  $A$  within a polynomial time  $p$ . Then for the control sequence  $\mathcal{C}_{A, p}$  one has

$$\mathcal{C}_{A, p} = o(1/n^k)$$

for any positive integer  $k$ . Due to the inequalities above, the control sequence for  $A \circ f$  with respect to the polynomial bound  $p \circ \mathcal{S}_f + T_f$  satisfies the following inequality

$$\mathcal{C}_{A \circ f, p \circ \mathcal{S}_f + T_f} = o(1/\mathcal{S}_f(n)^k) \leq o(1/n^k).$$

Hence  $(D, \{\mu_n\}_{n=1}^\infty)$  is SGPTIME decidable by  $A \circ f$ , as claimed.  $\square$

By Theorem 4.7 a CS-reduction generally increases time complexity and improves control sequence.

**4.2. Change of measure.** In this section, we define *change of measure* (CM) reductions.

**Definition 4.8.** Let  $(D, \mu), (E, \nu) \in \mathbf{DistNP}$  and  $D \xrightarrow{f} E$  a PTIME reduction such that

- $|x| = |f(x)|$  for any  $x \in \Sigma_D^*$ ;
- there exists a polynomial  $d$  such that for each  $x \in \Sigma_D^*$

$$\nu_{|x|}(f(x)) \geq \frac{\mu_{|x|}(x)}{d(|x|)}.$$

Then  $f$  is called a *CM-reduction*.

Figure 2 depicts the situation under consideration.

**Theorem 4.9** (CM-reduction is an SGPTIME reduction). *Let  $(D, \mu), (E, \nu) \in \mathbf{DistNP}$  and  $D \xrightarrow{f} E$  a CM-reduction. Then the following holds.*

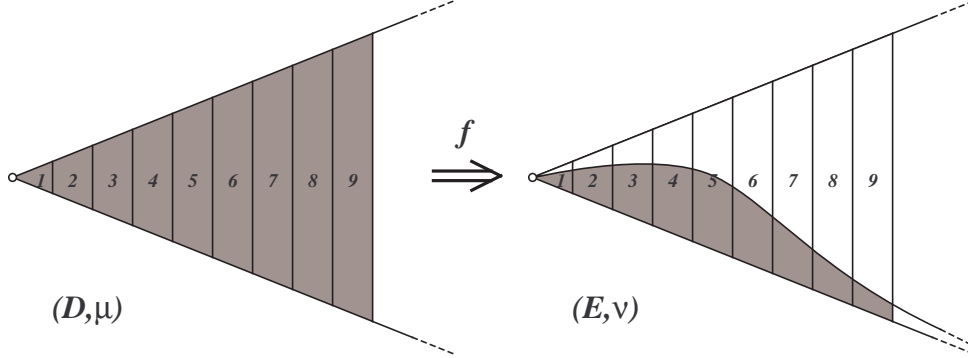


FIGURE 2. Scheme of a CM-reduction. A function  $f$  maps a distributional decision problem  $(D, \mu)$  into a distributional decision problem  $(E, \nu)$  so that the  $i$ th sphere in  $D$  is mapped exactly into the  $i$ th sphere in  $E$ . The grey part of  $E$  depicts the image of  $D$ .

- (a) If  $(E, \nu)$  is decidable by a TM  $A$  within a generic polynomial time bound  $p$  and  $C_{A,p} = o(1/d(k))$  (where  $d(k)$  is the function from Definition 4.8) then  $A \circ f$  GPtime decides  $(D, \mu)$ .
- (b)  $f$  is an SGPTIME reduction.

*Proof.* (a) Let  $A$  be an algorithm that generically decides  $(E, \nu)$  within a polynomial time upper bound  $p(m)$ . Then  $A \circ f$  is a partial decision algorithm for  $(D, \mu)$ . Recall, that  $f$  preserves the size. Therefore,  $A \circ f$  decides  $D$  within the polynomial time bound  $p + T_f$  everywhere, except, maybe, a subset

$$\{x \in D \mid T_A(f(x)) > p(|f(x)|) = p(|x|)\}.$$

To prove the statement it suffices to show that the set above is generic in  $(\Sigma_D^*, \mu)$ .

$$\begin{aligned} & \mu_k \{x \in D \mid T_A(f(x)) > p(|f(x)|), |x| = k\} \\ & \leq \nu_k \{f(x) \in E \mid T_A(f(x)) > p(|x|), |x| = k\} d(k) \\ & \leq \nu_k \{y \in E \mid T_A(y) > p(|y|), |y| = k\} d(k) \\ & = C_{A,p}(k) d(k) = o(1). \end{aligned}$$

(b) If an algorithm  $A$  SGPTIME decides  $(E, \nu)$  then  $C_{A,p} = o(1/q(k))$  for any polynomial  $q$ . Therefore, by part 1),  $C_{A \circ f, p+T_f} \leq o(d(k)/q(k))$  for every polynomial  $q$ . In particular, for  $q = dq'$  one has

$$C_{A \circ f, p+T_f} \leq o(d(k)/q(k)) = o(1/q'(k))$$

for any polynomial  $q'$ , as required.  $\square$

**4.3. Reduction to a problem with the binary alphabet.** In this section we show that each **DistNP** problem over a finite alphabet  $\Sigma$  can be reduced to a **DistNP** problem over a binary alphabet  $\{0, 1\}$ .

**Theorem 4.10.** *Let  $(D, \mu)$  be an **DistNP** problem over a finite alphabet  $\Sigma$ . Then there exists a **DistNP** problem  $(E, \nu)$  over the binary alphabet  $\{0, 1\}$  and a CS-reduction  $D \xrightarrow{f} E$  with linear size function  $\mathcal{S}_f$ .*

*Proof.* Suppose that  $\Sigma = \{a\}$  is an one-letter alphabet. Let  $f : \{a\}^* \rightarrow \{0, 1\}^*$  be a monoid homomorphism defined by  $f(a) = 0$ . Put  $E^+ = f(D^+)$  and  $E = (\{0, 1\}^*, E^+)$ . Define a spherical ensemble of measures  $\nu$  on  $\{0, 1\}^*$  to be

$$\nu_{|y|}(y) = \sum_{f(x)=y} \mu_{|x|}(x).$$

Clearly,  $(E, \nu) \in \mathbf{DistNP}$  and  $f$  is a CS-reduction with linear size-growth function  $\mathcal{S}_f$ . By Theorem 4.7,  $f$  is an SGTime reduction.

Suppose that  $|\Sigma| = n$ , where  $n \geq 3$ . Define a function  $f$  as follows. Put  $f(\varepsilon) = \varepsilon$  and, if  $|x| \geq 1$  and  $x$  is the  $k$ th element in  $\Sigma^n$  (in the lexicographical order), then  $f$  maps  $x$  into the  $k$ th element of  $\{0, 1\}^{\lceil |x| \log_2 n \rceil}$ . As above, we put  $E^+ = f(D^+)$ . Let  $\nu$  be the  $f$ -transfer of  $\mu$ . The problem  $(E, \nu)$  belongs to  $\mathbf{DistNP}$  because  $(D, \mu) \in \mathbf{DistNP}$  and  $f$  is a Ptime reduction. Clearly,  $f$  is a CS-reduction with a linear size-growth function  $\mathcal{S}_f(i) = \lceil i \log_2 n \rceil$ . By Theorem 4.7,  $f$  is an SGTime reduction.  $\square$

**4.4. On restrictions of problems.** Let  $D = (\Sigma_D^*, D^+)$  be a problem and  $S \subseteq \Sigma_D^*$ . In this section we consider the restriction  $D_S$  of  $D$  to the subset  $S$ . Intuitively,  $D_S$  is the same problem as  $D$ , only the set of inputs is restricted to  $S$ . The most natural formalization of  $D_S$  would be  $(S, D^+ \cap S)$ , allowing the domain  $S$  not equal to  $\Sigma_D^*$ , contrary to our assumption on algorithmic problems. In this case one can stratify the domain  $I$  as a union  $I = \cup_{n=0}^{\infty} I_n$ , where  $I_n = I \cap \Sigma^n$ , and leave only those  $I_n$  that are non-empty. Then, one can obtain an ensemble of measures  $\mu' = \{\mu'_n\}_{n=0}^{\infty}$  on  $I$  relative to the stratification above, where  $\mu'_n$  is the measure on  $I_n$  induced by  $\mu_n$ . After that, the theory of distributional problems of this type can be developed similarly to the one already considered. However, it is a bit awkward and heavier in notation. We choose another way around this problem – we change the ensemble of measures, but do not change the input space.

Let  $(D, \mu)$  be a distributional problem. For a subset  $S \subseteq \Sigma_D^*$  consider the ensemble of probability measures  $\mu^S$  on  $\Sigma_D^*$   $S$ -induced by  $\mu$  (see Section 3.1). The distributional problem  $(D, \mu^S)$  is called the *restriction* of the distributional problem  $(D, \mu)$  to the subset  $S$ .

**Lemma 4.11.** *Let  $(D, \mu) \in \mathbf{DistNP}$  and  $S \subseteq \Sigma_D^*$ . If the function  $n \rightarrow \mu_n(S \cap \Sigma_D^n)$  is Ptime computable then  $(D, \mu^S) \in \mathbf{DistNP}$ .*

*Proof.* Follows immediately from Lemma 3.8.  $\square$

**Lemma 4.12.** *Let  $(D, \mu) \in \mathbf{DistNP}$ ,  $S \subseteq \Sigma_D^*$ , and  $(D, \mu^S) \in \mathbf{DistNP}$ . If an algorithm  $A$  GPtime decides  $(D, \mu)$  with a control sequence  $q_i$  such that the sequence*

$$c_i = \begin{cases} q_i / \mu_i(S \cap \Sigma_D^i), & \text{if } \mu_i(S \cap \Sigma_D^i) \neq 0; \\ q_i, & \text{if } \mu_i(S \cap \Sigma_D^i) = 0. \end{cases}$$

*converges to 0, then  $A$  GPtime decides  $(D, \mu^S)$  with the control sequence bounded from above by  $\{c_i\}_{i=1}^{\infty}$ .*

*Proof.* Let  $p(n)$  be a generic polynomial time upper bound of the algorithm  $A$  and  $F = \{x \in \Sigma_D^* \mid T_A(x) > p(|x|)\}$ . Set  $S_i = S \cap \Sigma_D^i$ ,  $F_i = F \cap \Sigma_D^i$ . Then  $q_i = \mu_i(F_i)$  and for  $i \in \mathbb{N}$  one has

$$\mu_i^S(F_i) = \begin{cases} \frac{\mu_i(F_i \cap S_i)}{\mu_i(S_i)}, & \text{if } \mu_i(S_i) \neq 0; \\ \mu_i(F_i), & \text{if } \mu_i(S_i) = 0. \end{cases}$$

Hence

$$\mu_i^S(F_i) \leq \begin{cases} \frac{q_i}{\mu_i(S_i)}, & \text{if } \mu_i(S_i) \neq 0; \\ q_i, & \text{if } \mu_i(S_i) = 0. \end{cases}$$

Thus, the sequence  $\{\mu_i^S(F_i)\}_{i=0}^\infty$  converges to 0.  $\square$

**Corollary 4.13.** Let  $(D, \mu) \in \mathbf{DistNP}$ ,  $S \subseteq \Sigma_D^*$ , and  $(D, \mu^S) \in \mathbf{DistNP}$ . If there exists a polynomial  $d$  such that  $\mu_i(S \cap \Sigma_D^i) \geq 1/d(i)$  for  $\mu_i(S \cap \Sigma_D^i) \neq 0$  then the identity function  $id : \Sigma_D^* \hookrightarrow \Sigma_D^*$  gives an SGPtime reduction

$$(D, \mu^S) \xrightarrow{id} (D, \mu).$$

**Remark 4.14.** We would like to emphasize that the situation with restrictions of problems in **GP** is quite different from the “average-case” one, where almost any restriction preserves the property of being polynomial time computable on average.

## 5. DISTRIBUTIONAL BOUNDED HALTING PROBLEM

In this section we, following [12], define the distributional bounded halting problem and prove that it is SGPtime complete in **DistNP**.

Let  $M$  be a nondeterministic Turing machine with the binary tape alphabet  $\Sigma = \{0, 1\}$ . Intuitively, the bounded halting problem for  $M$  is the following algorithmic question:

For a positive integer  $n$  and a binary string  $w$  such that  $|w| < n$  decide if there is a halting computation for  $M$  on  $w$  in at most  $n$  steps.

By our definitions (see Section 2.1) instances of algorithmic problems are words (not pairs of words) in some alphabet, so to this end we encode a pair  $(n, w)$  by the binary string  $c(n, w) = 1^m 0 w$  such that  $n = |1^m 0 w|$ . Notice, that any binary string containing 0 is the code  $c(n, w)$  for some  $n, w$ . Denote by  $BH(M)^+$  the subset of all binary strings  $c(n, w)$ , where  $n \in \mathbb{N}, w \in \Sigma^*$ , such that  $M$  halts on  $w$  within  $n$  steps. From now on we refer to the problem  $BH(M) = (\Sigma^*, BH(M)^+)$  as the *bounded halting problem*.

To turn  $BH(M)$  into a distributional problem we introduce a spherical ensemble  $\nu = \{\nu_n\}_{n=1}^\infty$  of probability measures as follows. For  $u \in \{0, 1\}^*$  put

$$\nu_{|u|}(u) = \begin{cases} \frac{1}{|u|2^{|w|}}, & \text{if } u = 1^m 0 w; \\ 1, & \text{if } u = \varepsilon; \\ 0, & \text{if } u = 1^k \text{ for some } k \geq 1. \end{cases}$$

The problem  $(BH(M), \nu)$  is the *distributional halting problem* for  $M$ , we refer to it as  $DBH(M)$ .

A positive integer  $n$  is called *longevous* for an input  $w$  of an NTM  $M$  if every halting computation of  $M$  on  $w$  has at most  $n$  steps. A function  $g(n)$  is a *longevity guard* for  $M$  if for every input  $w$  the number  $g(|w|)$  is longevous for  $w$ . Notice, that if  $g$  is a longevity guard for  $M$ , then any function  $h \geq g$  is also a longevity guard for  $M$ . In what follows, we always assume that a longevity guard satisfies the following conditions:

- (L1)  $g(|w|) \geq |w|$ ;
- (L2)  $g(|w|)$  is strictly increasing.

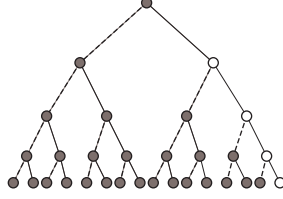


FIGURE 3. Probability space for halting problem (a finite part). Dashed lines correspond to 0, solid lines correspond to 1. Each dot is associated with the label of the path from the root to itself. Grey dots have non-trivial measure.

**Remark 5.1.** For every problem  $D \in \mathbf{NP}$ , there is an NTM  $D$  that decides  $D$  and has a polynomial longevity guard  $g(n)$ , satisfying the conditions (1), (2) above.

Since  $M$  halts on an input  $w$  if and only if it halts on  $w$  within  $g(|w|)$  steps, there is no much use to consider instances  $(n, w)$  of the halting problem for  $M$  with  $n > g(|w|)$ . A rigorous formalization of this observation is to restrict the problem  $DBH(M)$  to the subset of instances

$$C(g) = \{c(g(|w|), w) \mid w \in \Sigma^*\}.$$

More generally, for a computable function  $g(n)$ , satisfying conditions (1) and (2), consider the set  $C(g)$  as above and denote by  $\nu = \nu(g)$  the ensemble of measures for  $\Sigma^*$  which is  $C(g)$ -induced by  $\nu$ . Let  $DBH(M, g) = (BH(M), \nu_g)$  be the restriction of the problem  $DBH(M)$  to  $C(g)$ .

**Proposition 5.2.** Let  $M$  be an NTM and  $g : \mathbb{N} \rightarrow \mathbb{N}$  a polynomial function. Then  $DBH(M, g) \in \mathbf{DistNP}$  and the identity function  $id : \Sigma^* \rightarrow \Sigma^*$  gives an SGptime reduction  $DBH(M, g) \xrightarrow{id} DBH(M)$ .

*Proof.* Observe first that the function  $n \rightarrow \nu_n(C(g) \cap \Sigma^n)$  is Ptime computable. Indeed, if  $u = 1^m 0 w \in C(g) \cap \Sigma^n$ , then  $n = g(|w|)$ , so  $|w| = g^{-1}(n) = k$  is uniquely defined (since  $g$  is monotone). In this case,  $\nu_n(u) = \frac{1}{n2^k}$  depends only on  $n$ , hence

$$\nu_n(C(g) \cap \Sigma^n) = \frac{1}{n2^k} |C(g) \cap \Sigma^n| = \frac{1}{n2^k} 2^k = \frac{1}{n}.$$

Therefore,

$$(3) \quad \nu_n(C(g) \cap \Sigma^n) = \begin{cases} \frac{1}{n}, & \text{if } g^{-1}(n) \neq \emptyset; \\ 0, & \text{otherwise.} \end{cases}$$

Since the function  $g$  is polynomial it takes at most  $O(ng(n))$  time to check if  $g^{-1}(n) = \emptyset$  or not. Now, by Lemma 4.11  $DBH(M, g) \in \mathbf{DistNP}$ . Equalities 3 and Corollary 4.13 imply that  $DBH(M, g) \xrightarrow{id} DBH(M)$  is an SGptime reduction, as claimed.  $\square$

In the proofs below, we use the following encoding of natural numbers and Turing machines. Let a string  $b = b_k \dots b_0$  be a binary expansion for  $n \in \mathbb{N}$ , i.e.,  $n = \sum_{i=0}^k b_i 2^i$ ,  $b_k = 1$ , and  $b_0, \dots, b_{k-1} \in \{0, 1\}$ . Denote by  $\bar{n}$  the binary string

$$1b_k 1b_{k-1} \dots 1b_0$$

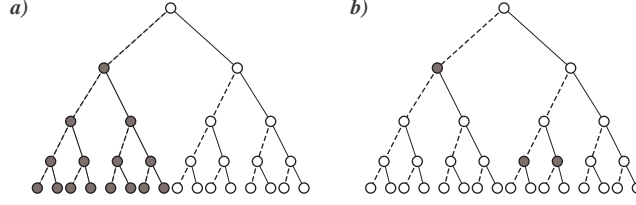


FIGURE 4. Examples of restricted problems  $BH(M, g)$ , where a)  $g(n) = n + 1$  and b)  $g(n) = 2n + 1$ . Only grey dots have non-trivial probability.

obtained from  $b$  by inserting 1 in front of each symbol in  $b$ . Let  $\gamma : M \rightarrow \gamma(M)$  be a polynomial time computable enumeration of (nondeterministic) Turing machines, such that  $\gamma(M)$  is a binary representation of a natural number and every natural number is equal to  $\gamma(M)$  for some  $M$ . Denote by  $\overline{M}$  the string  $\gamma(M)$ .

**Theorem 5.3.** *For any  $(D, \mu) \in \mathbf{DistNP}$  there exists an NTM  $M$  over the binary alphabet  $\Sigma = \{0, 1\}$ , a polynomial longevity guard  $g$  for  $M$ , and an SGPtime reduction  $(D, \mu) \xrightarrow{f} DBH(M, g)$ .*

*Proof.* Fix  $(D, \mu) \in \mathbf{DistNP}$ . We divide the proof of the theorem into two parts. First, we construct a Turing machine  $M$ , a longevity guard  $g$  for  $M$ , and a Ptime reduction  $f$  from the original problem  $(D, \mu)$  to  $BH(M, g)$ . Then we show that  $f$  is a composition of CS and CM reductions defined in Sections 4.1 and 4.2 and, hence,  $f$  is an SGPtime reduction  $(D, \mu) \xrightarrow{f} DBH(M, g)$ .

**Part I.** By Theorem 4.10 we may assume that the alphabet of  $D$  is binary. Now, since  $D \in \mathbf{NP}$  there exists an NTM  $A_D$  such that:

- $A_D$  has a halting computation on an input  $w$  if and only if  $w \in D$ ;
- $A_D$  has a polynomially bounded longevity guard.

Recall that  $\mu^*$  is the ensemble of probability distributions

$$\mu_{|x|}^*(x) = \mu_{|x|}(\{y \in \Sigma^{|x|} \mid y <_{lex} x\}).$$

Since  $(D, \mu) \in \mathbf{DistNP}$  the ensemble  $\mu^* = \{\mu_n^*\}$  is Ptime computable. For  $x \in \{0, 1\}^*$  define a function

$$\hat{\mu}_{|x|}(x) = \begin{cases} \mu_{|x|}^*(x^+), & \text{if } x \neq 1^{|x|}; \\ 1, & \text{if } x = 1^{|x|}; \end{cases}$$

where  $x^+$  is the lexicographic successor of  $x$ . For  $x \in \{0, 1\}^*$  such that  $\mu_{|x|}(x) > 2^{-|x|}$  define  $x' = x_0 \dots x_k$  to be the smallest (in shortlex ordering) binary string such that

$$\mu_{|x|}^*(x) < x_0.x_1x_2\dots x_k1 \leq \hat{\mu}_{|x|}(x),$$

where  $x_0.x_1x_2\dots x_k$  is the binary expansion of a real number in the interval  $[0, 1]$ . One can describe  $x'$  as follows. Assume first that  $\hat{\mu}_{|x|}(x) \neq 1$ . Since  $\mu_{|x|}(x) > 2^{-|x|}$  the binary expansions of  $\mu_{|x|}^*(x)$  and  $\hat{\mu}_{|x|}(x)$  differ within the first  $|x| - 1$  bits after “.”, i.e.,

$$\mu_{|x|}^*(x) = 0.x_1x_2\dots x_k0\dots \quad \text{and} \quad \hat{\mu}_{|x|}(x) = 0.x_1x_2\dots x_k1\dots$$



for some  $k \leq |x| - 1$ . In this case  $x' = 0x_1x_2 \dots x_k$ . The case  $\hat{\mu}_{|x|}(x) = 1$  is similar. It follows that for every  $x \in \Sigma^*$  such that  $\mu_{|x|}(x) > 2^{-|x|}$  we have  $|x'| \leq |x|$  and

$$x_0.x_1x_2 \dots x_k 1 - 2^{-|x'|} \leq \mu_{|x|}^*(x) < \hat{\mu}_{|x|}(x) < x_0.x_1x_2 \dots x_k 1 + 2^{-|x'|}.$$

Hence,  $\mu_{|x|}(x) < 2 \cdot 2^{-|x'|}$ . Define

$$x'' = \begin{cases} 0x, & \text{if } \mu_{|x|}(x) \leq 2^{-|x|}; \\ 1x', & \text{if } \mu_{|x|}(x) > 2^{-|x|}. \end{cases}$$

Notice that for every  $x \in \Sigma^*$ ,  $\mu_{|x|}(x) \leq 4 \cdot 2^{-|x''|}$  and  $|x''| \leq |x| + 1$ .

Now we describe an NTM  $M$ , a function  $g$ , and a reduction  $f : (D, \mu) \rightarrow BH(M, g)$ . If  $M$  is defined and  $g$  is a polynomial longevity guard for  $M$ , then the reduction  $f$  is defined for  $x \in \Sigma^*$  by

$$(4) \quad f(x) = 1^{g(|x|)-|0\bar{n}0x''|} 0\bar{n}0x''$$

where  $n = |x|$ . It is left to define  $M$  and  $g$ .

The machine  $M$  on a binary input  $u$  executes the following algorithm:

- A. If  $u$  is not in the form  $1^k 0\bar{n}0bw$ , where  $b \in \Sigma$  and  $w \in \Sigma^*$ , then loop forever.
- B. If  $u = 1^k 0\bar{n}0bw$  then decode  $n$ .
- C. If  $b = 0$ :
  - (1) if  $\mu_n(w) > 2^{-|w|}$  then loop forever;
  - (2) otherwise simulate  $A_D$  on  $w$ .
- D. If  $b = 1$ :
  - (1) find the lexicographic smallest  $x \in \{0, 1\}^n$  satisfying  $\mu_n^*(x) < 0.w1 \leq \hat{\mu}_n(x)$  using divide and conquer approach;
  - (2) if  $\mu_n(x) \leq 2^{-|x|}$  or  $x' \neq w$  then loop forever;
  - (3) otherwise simulate  $A_D$  on  $x$ .

By construction,  $M$  has a halting computation on  $u \in \Sigma^*$  if and only if  $u = f(x)$  for some  $x \in \Sigma^*$  and  $x \in D^+$ .

We claim that  $M$  has a polynomial longevity guard  $g$ . Indeed, since  $D \in \mathbf{NP}$  it follows that an NTM  $A_D$  has a polynomial longevity guard, and all steps in the algorithm above, except simulation of  $A_D$ , can be performed by deterministic polynomial time algorithms. Therefore,  $M$  has a polynomial longevity guard  $g$ , as claimed. In particular,  $D \xrightarrow{f} BH(M, g)$  is a Ptime reduction, as claimed.

**Part II.** Now we prove that  $f$  is an SGPTIME reduction. We start with the following lemma.

**Lemma 5.4.** *For every  $m \in \mathbb{N}$  and  $x \in \{0, 1\}^*$  the following inequality holds:*

$$(5) \quad \nu_{|f(x)|}(f(x)) \geq \frac{1}{16|x|^2g(|x|)} \cdot \mu_{|x|}(x).$$

*Proof.* For every  $x \in \{0, 1\}^*$  there are two possibilities. If  $\mu_{|x|}(x) \leq 2^{-|x|}$ , then  $f(x) = 1^{g(|x|)-|0\bar{n}00x|} 0\bar{n}00x$  and its measure is:

$$\begin{aligned} \nu_{|f(x)|}(f(x)) &= \frac{1}{g(|x|)2^{|x|+2\lceil \log_2 |x| \rceil + 2}} \\ &\geq \frac{1}{g(|x|)2^{|x|+2\log_2 |x|+3}} = \frac{1}{8|x|^2g(|x|)2^{|x|}} \\ &\geq \frac{1}{8|x|^2g(|x|)} \cdot \mu_{|x|}(x) \geq \frac{1}{8|x|^2g(|x|)} \cdot \mu_{|x|}(x). \end{aligned}$$

If  $\mu_{|x|}(x) > 2^{-|x|}$ , then  $f(x) = 1^{g(|x|)-|0\overline{m}01x'|}0\overline{m}01x'$  and its measure is:

$$\begin{aligned}\nu_{|f(x)|}(f(x)) &= \frac{1}{g(|x|)2^{|x'|+2\lceil\log_2|x'|\rceil+2}} \\ &\geq \frac{1}{g(|x|)2^{|x'|+2\log_2|x'|+3}} = \frac{1}{8|x'|^2g(|x|)2^{|x'|}} \\ &\geq \frac{1}{16|x|^2g(|x|)} \cdot \mu_{|x|}(x)\end{aligned}$$

since  $|x'| \leq |x|$  and  $\mu_{|x|}(x) < 2 \cdot 2^{-|x'|}$ . In each case the inequality (5) holds.  $\square$

By construction of  $g$ , all elements of  $\{0,1\}^n$  are mapped to elements of size  $g(n)$ , hence,  $f$  is size-invariant. It follows that a function  $f$  is a composition of a CS-reduction with the polynomial size-growth function  $\mathcal{S}_f(n) = g(n)$  and a CM-reduction with a polynomial density function  $\frac{1}{16|x|^2g(|x|)}$ . Thus,  $f$  is an SGPtime reduction.  $\square$

Let  $U$  be a universal NTM such that:

- (a)  $U$  accepts inputs of the form  $\overline{M}0w$ , where  $\overline{M}$  is the encoding of an NTM  $M$  over a binary alphabet and  $w \in \Sigma^*$ ;
- (b)  $U$  simulates  $M$  on  $w$ , i.e.,  $M$  halts on  $w$  if and only  $U$  halts on  $\overline{M}0w$ , in which case they both have the same answer (the same final configurations);
- (c)  $U$  has a polynomial-time slow-down, i.e., there exists a polynomial function  $s(k)$  such that  $T_M(w) \geq T_U(\overline{M}0w)/s(|w|)$ .

See, for example, [22] on how such a deterministic Turing machine  $U$  can be constructed, a nondeterministic one can be constructed in a similar way.

**Theorem 5.5.** *For every NTM  $M$  over a binary alphabet  $\Sigma$ , there exists a Ptime computable function  $h : \mathbb{N} \rightarrow \mathbb{N}$  and an SGPtime reduction  $DBH(M) \xrightarrow{f} DBH(U, h)$ .*

*Proof.* Let  $M$  be an NTM over  $\Sigma$  and  $g$  a polynomial longevity guard for  $M$ . Define (in the notation above) a function  $f : \Sigma^* \rightarrow \Sigma^*$  by

$$f(x) = 1^{g(|x|)s(|x|)-|0\overline{m}0\overline{M}0x''|}0\overline{m}0\overline{M}0x'',$$

where  $s$  is the polynomial from the description of the machine  $U$  above. Put  $h(n) = g(n)s(n)$ . Clearly,  $f$  gives a Ptime reduction  $BH(M) \xrightarrow{f} BH(U, h)$ . To show that  $f$  is an SGPtime reduction, one can argue as in the proof of Theorem 5.3. To carry over the argument, one needs the following inequality for every  $m \in \mathbb{N}$  and  $x \in \{0,1\}^*$ :

$$\nu_{|f(x)|}(f(x)) \geq \frac{1}{16|x|^2g(|x|)s(|x|)2^{|\overline{M}|+1}} \cdot \mu_{|x|}(x)$$

which differs from the inequality (5) by a polynomial factor  $2^{|\overline{M}|+1}s(|x|)$  in the denominator. The proof of this is similar to the one in Lemma 5.4 and we omit it.  $\square$

**Corollary 5.6.** There exists an NTM  $U$  such that  $DBH(U)$  is SGPtime complete.

*Proof.* By Theorem 5.3 for any **DistNP** problem  $(D, \mu)$  there exists an NTM  $M$  over a binary alphabet  $\Sigma$ , a polynomial longevity guard  $g$  of  $M$ , and an SGPtime reduction of  $(D, \mu)$  to  $DBH(M, g)$ . By Proposition 5.2, there is an SGPtime reduction of  $DBH(M, g)$  to  $DBH(M)$ . By Theorem 5.5, there exists a Ptime computable

function  $h$  and an SGPtime reduction  $DBH(M) \xrightarrow{f} DBH(U, h)$ . Now, again by Proposition 5.2, there is an SGPtime reduction of  $DBH(U, h)$  to  $DBH(U)$ . Hence,  $(D, \mu)$  is SGPtime reducible to  $DBH(U)$ , as claimed.  $\square$

## 6. OPEN PROBLEMS

In this section we discuss some open problems on generic complexity.

**Problem 6.1.** Is it true that every **NP**-complete problem is generically in **P**?

In fact, even a much stronger version of the question above is still open:

**Problem 6.2.** Is it true that every **NP**-complete problem is strongly generically in **P**?

Some of the well-known **NP**-complete problems are in **GP**, or in **SGP**, see [7] for examples. However, there is no general approach to this problem at present. If the answer to one of the questions above (in particular, the second one) is affirmative, then it will imply that for all practical reasons **NP**-complete problems are rather easy. Otherwise, we will have an interesting partition of **NP**-complete problems into several classes with respect to their generic behavior.

It was shown in [13] that the halting problem for one-end tape Turing machines is in **GP**. It remains to be seen if a similar result holds for Turing machines where the tape is infinite at both ends.

**Problem 6.3.** Is it true that the halting problem is in **GP** for Turing machines with one tape that is infinite at both ends?

It is known (see [7]) that the classes of functions that are polynomial on average and generically polynomial are incompatible, i.e., none of them is a subclass of the other. Nonetheless, the relationship between **SGP**-complete and **NP**-complete on average is still unclear. To this end, the following problem is of interest.

**Problem 6.4.** Is it true that every **NP**-complete on average problem is **SGP**-complete?

## REFERENCES

- [1] G. Arzhantseva. Generic properties of finitely presented groups and Howson's theorem. *Comm. Algebra*, 26:3783–3792, 1998.
- [2] L. Bienvenu, Day A., and R. Holzl. From bi-immunity to absolute undecidability. *J. Symbolic Logic*, 78:1218–1228, 2013.
- [3] A. V. Borovik, A. G. Myasnikov, and V. N. Remeslennikov. Multiplicative measures on free groups. *Int. J. Algebra Comput.*, 13:705–731, 2003.
- [4] C. Champetier. Propriété statistiques des groupes de présentation finie. *Adv. in Math.*, 116:197–262, 1995.
- [5] R. Downey, C. Jockusch, T. McNicholl, and P. Schupp. Asymptotic density and the Ershov hierarchy. To appear. Available at <http://arxiv.org/abs/1309.0137>, 2014.
- [6] R. Downey, C. Jockusch, and P. Schupp. Asymptotic density and computably enumerable sets. *J. Math. Log.*, 13:43, 2013.
- [7] R. Gilman, A. G. Myasnikov, A. D. Miasnikov, and A. Ushakov. Generic complexity of algorithmic problems. In preparation.
- [8] R. Gilman, A. G. Myasnikov, A. D. Miasnikov, and A. Ushakov. Report on generic case complexity. Preprint, available at <http://arxiv.org/abs/0707.1364>.
- [9] M. Gromov. Hyperbolic groups. In *Essays in group theory*, volume 8 of *MSRI Publications*, pages 75–263. Springer, 1985.

- [10] M. Gromov. Asymptotic invariants of infinite groups. In *Geometric Group Theory II*, volume 182 of *LMS lecture notes*, pages 290–317. Cambridge Univ. Press, 1993.
- [11] M. Gromov. Random walks in random groups. *Geom. Funct. Analysis*, 13:73–146, 2003.
- [12] Y. Gurevich. Average case completeness. *J. Comput. Syst. Sci.*, 42:346–398, 1991.
- [13] J. D. Hamkins and A. G. Miasnikov. The halting problem is decidable on a set of asymptotic probability one. *Notre Dame Journal of Formal Logic*, 47:515–524, 2006.
- [14] G. Igusa. Nonexistence of minimal pairs for generic computability. *J. Symbolic Logic*, 78(2):511–522, 2013.
- [15] R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95)*, pages 134–147, 1995.
- [16] C. Jockusch and P. Schupp. Generic computability, Turing degrees, and asymptotic density. *J. Lond. Math. Soc.*, 85(2):472–490, 2012.
- [17] I. Kapovich, A. G. Miasnikov, P. Schupp, and V. Shpilrain. Generic-case complexity, decision problems in group theory and random walks. *J. Algebra*, 264:665–694, 2003.
- [18] I. Kapovich, A. Miasnikov, P. Schupp, and V. Shpilrain. Average-case complexity and decision problems in group theory. *Adv. Math.*, 190:343–359, 2005.
- [19] L. Levin. Average case complete problems. *SIAM J. Comput.*, 15:285–286, 1986.
- [20] A. G. Miasnikov and A. Rybalov. On generically undecidable problems. in preparation.
- [21] A. G. Miasnikov, V. Shpilrain, and A. Ushakov. *Non-Commutative Cryptography and Complexity of Group-Theoretic Problems*. Mathematical Surveys and Monographs. AMS, 2011.
- [22] T. Neary and D. Woods. Small fast universal Turing machines. technical report NUIM-CS-TR-200511, National university of Ireland, Maynooth, 2005.
- [23] A. Yu. Ol’shanskii. Almost every group is hyperbolic. *Int. J. Alg. Comput.*, 2:1–17, 1992.
- [24] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [25] J. Wang. Average-case completeness of a word problem for groups. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, STOC ’95, pages 325–334. ACM, 1995.

DEPARTMENT OF MATHEMATICS, STEVENS INSTITUTE OF TECHNOLOGY, HOBOKEN, NJ, USA  
*E-mail address*: amiasnik,aushakov@stevens.edu